УДК 004.431.2

А. А. Михайлов

Институт динамики систем и теории управления СО РАН
ул. Лермонтова, 134, Иркутск, 664003, Россия

E-mail: mikhailov@icc.ru

АНАЛИЗ ГРАФА ПОТОКОВ УПРАВЛЕНИЯ В ЗАДАЧЕ ДЕКОМПИЛЯЦИИ ПОДПРОГРАММ ОБЪЕКТНЫХ ФАЙЛОВ DCUIL

Рассматривается одна из задач декомпиляции – реконструкция операторов высокоуровневых языков программирования на примере объектных файлов *dcuil*.

Ключевые слова: декомпиляция, CIL, dcuil.

Введение

Для разработки большинства сложных программных систем в среде *Delphi* часто используются сторонние компоненты, предоставляемые в виде скомпилированных модулей. Такой подход существенно сокращает время и стоимость разработки программного обеспечения. С другой стороны, наличие сторонних модулей уменьшает надежность программного обеспечения с точки зрения информационной безопасности из-за возможного наличия уязвимостей, способствующих успешным атакам на информационную систему. Кроме того, сторонние компоненты могут содержать ошибки, исправление которых может оказаться затруднительным из-за невозможности связаться с разработчиком, отсутствием исходных кодов у разработчика (в связи с их утратой), и т. д. Также в некоторых случаях может потребоваться доработка сторонних модулей, без возможности использования исходных кодов.

Сторонние модули чаще всего распространяются в закрытом формате *dcu* (*Delphi compiled unit*). В ИДСТУ СО РАН А. Е. Хмельновым разработан инструмент *dcu32int* [1] для разбора модулей *Delphi*. Данный инструмент позволяет получить исходный код на языке ассемблера и интерфейсную часть модуля. Результат разбора в таком виде в отличие от программы на языке высокого уровня не предоставляет необходимого уровня абстракции для того, чтобы за приемлемое время и трудозатраты идентифицировать алгоритмические конструкции, а также отследить способы взаимодействия элементов программы, в то время как наличие восстановленной программы на языке высокого уровня повышает уровень абстракции, что существенно упрощает ее анализ.

Формат объектных файлов Delphi

В объектных файлах Delphi в отличие от исполняемого файла в формате *PE* программа оказывается более структурированной, например, выделены блоки памяти, соответствующие коду каждой процедуры; имеется информация о типах данных; может присутствовать отладочная информация. Такая информация отсутствует в обычных исполняемых файлах. В общем виде формат файла скомпилированных юнитов Delphi выглядит следующим образом:

[1] Программа для разбора юнитов Delphi DCU32INT. URL: http://hmelnov.icc.ru/DCU/index.ru.html

Михайлов А. А. Анализ графа потоков управления в задаче декомпиляции подпрограмм объектных файлов dcuil // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2014. Т. 12, вып. 2. С. 74–79.

ISSN 1818-7900. Вестник НГУ. Серия: Информационные технологии. 2014. Том 12, выпуск 2

сначала идет небольшой заголовок, в котором содержится общая информация о файле, такая как размер, время компиляции и т. д. После заголовка следует поток теговой информации. Для обобщения теги можно разделить на следующие группы:

- описания включаемых модулей и объектных файлов;
- импортируемых из этих модулей определений (типов данных, процедур, и т. д.);
- описания определений (типов данных, процедур и функций, и т. д.) из данного модуля;
- блок памяти, составленный из блоков для процедур и функций, образов констант, и т. д;
- информация для редактора связей (в какие места блока памяти необходимо занести адреса, получаемые из других модулей).
- отладочная информация.

Обзор методов восстановления высокоуровневых конструкций

Структурный анализ в задаче декомпиляции, как и в прямой задаче компиляции, основан на анализе графа потока управления [2]. Граф потоков управления состоит из базовых блоков, объединяющих в себе инструкции, которые гарантированно (с некоторыми оговорками [1]) выполняются последовательно. Дуги в графе соответствуют всем условным и безусловным командам передачи управления. На рис 1. представлен пример графа потока управления.

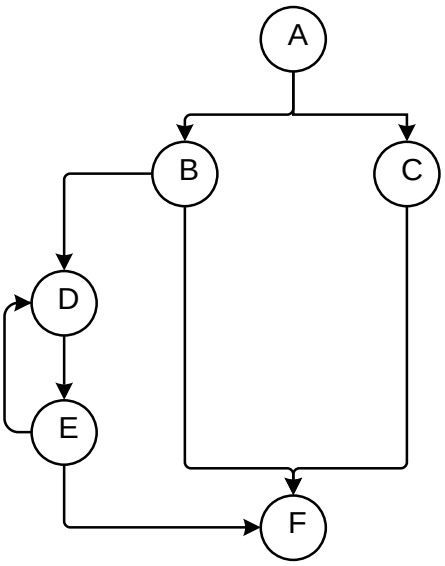

Рис. 1. Граф потоков управления

После того, как построен граф потока управления, начинается поиск высокоуровневых операторов. В статье [2] рассматривается два метода анализа графа потока управления.

Анализ дерева доминирующих вершин. В этом методе по графу потоков управления строится дерево доминирующих вершин. Вершина A доминирует над вершиной B, если любой путь к вершине B проходит через A. Наиболее эффективный алгоритм построения дерева доминаторов был предложен в статье [3]. Также в этой работе проведен обзор альтернативных алгоритмов.

После того как дерево доминаторов построено, на графе потока управления дуги помечаются как прямые, обратные и косые. Прямые дуги соединяют доминаторы и доминируемые вершины. На рис. 2. дуга, соединяющая узлы A и B, является прямой, потому что узел A доминирует над узлом B. Обратной дугой называется дуга, указывающая на предка, т. е. узел, который был пройден раньше в процессе обхода графа потоков управления. Все оставшиеся дуги помечаются как косые.

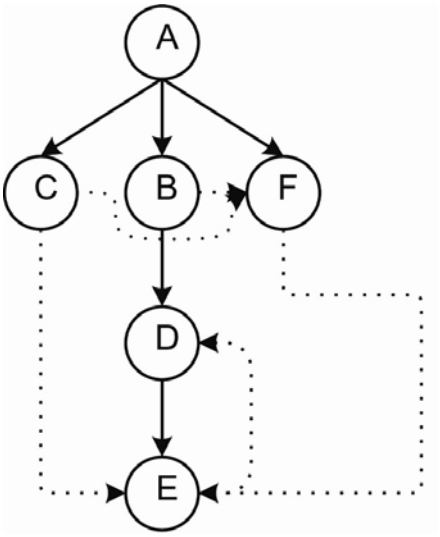

Рис. 1. Граф потоков управления

Размеченный граф потока управления (рис. 2) позволяет выделить только циклы, обратной дуге E → D соответствует цикл, состоящий из вершины D и всех вершин, доступных на пути из D в E.

Интервальный анализ. Интервальный анализ основан на семантически эквивалентных преобразованиях графа потока управления с помощью выделения регионов по заранее предопределенным шаблонам. В процессе обхода графа в порядке, обратном обходу в глубину, на каждую вершину накладываются шаблоны. В случае если

[2] http://ru.wikipedia.org/wiki/Граф потока управления

вершина является входной точкой циклической или ациклической конструкции, то регион, соответствующий заданной конструкции, выделяется в новую вершину, и дуги корректируются соответствующим образом. Тип выделяемого региона определяется путем сравнения подграфа рассматриваемой вершины с предопределенными шаблонами. Таким образом, граф преобразуется до одного региона, не содержащего дуг.

Реализация

В декомпиляторе *DCUIL2PAS* реализован метод восстановления высокоуровневых конструкций.

Граф потока управления в процессе анализа перестраивается в семантически эквивалентный граф регионов. При построении графа потоков управления каждому узлу добавляется метка со счетчиком ссылок (количество ссылок соответствует числу входящих дуг). При этом все инструкции условного и безусловного перехода приводятся к единому виду: **`if`** `x` **`op`** `y` **`then goto`** `label`. Затем строится дерево доминаторов и с его помощью выполняется разметка дуг графа на косые, обратные и прямые. После этого выполняется итеративный алгоритм выделения регионов. Шаблоны, используемые при выделении регионов, соответствуют высокоуровневым конструкциям: block, while, repeat, if-then-else, if-then, case. Подграфы, соответствующие заданным шаблонам, представлены на рис. 3.

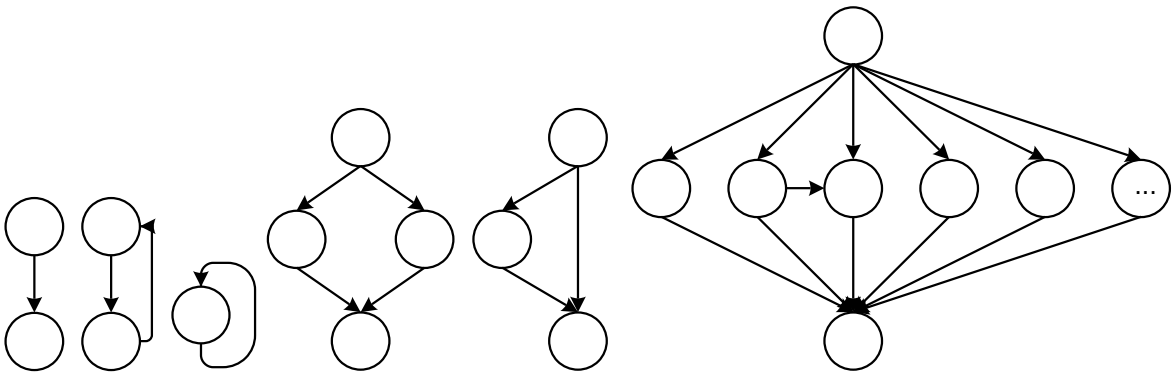

Рис. 3. Типы выделяемых регионов

Наложение шаблонов представляет собой обход дерева доминаторов в порядке, обратном обходу в ширину. Если рассматриваемый узел является вершиной одного из шаблонов, то все базовые блоки выделенного подграфа выделяются в новый регион.

Для каждого блока в зависимости от его вида уменьшаются счетчики ссылок для меток и удаляются выражения условных и безусловных переходов.

Отметим, что при выделении регионов не рассматриваются все косые дуги. После этого наложение выполняется со следующей непройденной вершины до тех пор, пока они все не будут пройдены.

Промежуточное представление реализовано с помощью иерархии классов (выражений), каждое из которых соответствует одному из шаблонов и является наследником базового класса `TCILExpr`. Этот класс реализует логику работы со счетчиками ссылок и задает набор обязательных методов:

- `Eval` – вычисляет значение выражения;
- `Eq(E: TCILExpr)` – возвращает **`true`**, если переданное выражение эквивалентно текущему;
- `AsString(BrRq: boolean)` – возвращает текстовое представление выражения;
- `Show` – выводит на печать текстовое представление выражения.

Например, шаблону `if-then-else` соответствует выражение вида `IfThenElse (Cond,True,False)`, где `Cond` – условие, `True,False` – регионы. Объединение сложных

условий для операторов ветвления происходит в процессе разбора выражений, полученных в результате анализа потоков управления. Выражение `IfThenElse(Cond, IfThenElse (Cond1,True1,False),False)` будет приведено к виду `IfThenElse(Cond and Cond1, True1, False)`.

Данный метод, в отличие от представленных в работах [4; 5], требует выполнения всего одной итерации за счет порядка обхода, основанного на порядке вершин в дереве доминаторов.

Предложенный метод находит не все высокоуровневые конструкции. Для отображения неструктурных переходов используются метки и оператор **`goto`**.

Пример декомпиляции файла в формате dcuil

Для демонстрации результатов работы описанных методов рассмотрим следующий небольшой пример. На рис. 4 представлен результат разбора функции, вычисляющей факториал, из файла Fact.dcuil, полученного с помощью программы *dcu32int*.

```
function Fact (n: Integer): Integer;
var
 Result: Integer; i: Integer; : Integer;
begin
 00: .   |02     | ldarg.0
 01: .   |17     | ldc.i4.1
 02: /.  |2F 04  | bge.s  $8
 04: .   |16     | ldc.i4.0
 05: .   |0C     | stloc.2
 06: +.  |2B 1A  | br.s  $22
 08: .   |17     | ldc.i4.1
 09: .   |0C     | stloc.2
 0A: .   |17     | ldc.i4.1
 0B: .   |02     | ldarg.0
 0C: .   |0A     | stloc.0
 0D: .   |0B     | stloc.1
 0E: .   |06     | ldloc.0
 0F: .   |07     | ldloc.1
 10: 2.  |32 10  | blt.s  $22
 12: .   |06     | ldloc.0
 13: .   |17     | ldc.i4.1
 14: X   |58     | add
 15: .   |0A     | stloc.0
 16: .   |08     | ldloc.2
 17: .   |07     | ldloc.1
 18: Z   |5A     | mul
 19: .   |0C     | stloc.2
 1A: .   |07     | ldloc.1
 1B: .   |17     | ldc.i4.1
 1C: X   |58     | add
 1D: .   |0B     | stloc.1
 1E: .   |07     | ldloc.1
 1F: .   |06     | ldloc.0
 20: 3ф  |33 F4  | bne.un.s  $16
 22: .   |08     | ldloc.2
 23: *   |2A     | ret
end ;
```

Рис. 4. Функция вычисления факториала на CIL

Исходный код в таком виде не предоставляет необходимого уровня абстракции для его анализа с приемлемыми трудозатратами. Для того чтобы восстановить код в язык высокого

уровня, необходимо знать в каждый момент выполнения состояние стека, значение аргументов и локальных переменных функции. Не говоря о том, что необходимо также знать семантику всех используемых опкодов CIL (Common Intermediate Language).

На рис. 5 приведен результат разбора функции из файла Fact.dcuil с помощью программы *DCUIL2PAS*. Результат разбора в таком виде, в отличие от результата, представленного на рисю 4, не содержит низкоуровневых инструкций, предоставляет более высокий уровень абстракции и является более привычным для анализа специалистом.

```
function  Fact (n: Integer): Integer;
var
  LocVar: integer;
  i: integer;
begin
  if (n < 1) then
     Result := 0
   else begin
    Result := 1;
    LocVar := n;
    i := 1;
    if (LocVar >= i) then begin
      LocVar := LocVar + 1;
      while (i <> LocVar) do begin
        Result := Result * i;
        i := i + 1;
      end;
    end;
  end;
end;
```

Рис. 5. Результат декомпиляции функции из рис. 4

Заключение

В работе предложен метод восстановления высокоуровневых конструкций в задаче декомпиляции объектных файлов *dcuil* на основе интервального анализа. Особенностью данного метода в отличие от предложенного в [5] является порядок обхода графа потоков управления, основанного на дереве доминаторов. Такой подход позволяет существенно сократить количество итераций наложения шаблонов.

Список литературы

1. *Ахо А., Лам М., Сети Р., Ульман Д. Д.* Компиляторы: принципы, технологии и инструменты. 2-е изд. М.: Вильямс, 2008. 1184 с.

2. *Cifuentes C., Simon D., Fraboulet A.* Assembly to High-Level Language Translation. Technical Report 439. Department of Computer Science and Electrical Engineering. The University of Queensland, 1998.

3. *Cooper K. D., Harvey T. J., Kennedy K.* A Simple, Fast Dominance Algorithm. Department of Computer Science. Rice University, 2001.

4. *Деревенец Е. О., Трошина Е. Н.* Структурный анализ в задаче декомпиляции // Прикладная информатика. 2009. № 4. С. 87–99.

5. *Cifuentes C.* Structuring Decompiled Graphs. Department of Computer Science, Univ. of Tasmania, 1996.

Материал поступил в редколлегию 23.05.2014

A. A. Mikhailov

CONTROL FLOW ANALYSIS IN THE TASK OF DECOMPILATION SUBROUTINES IN THE OBJECT FILES DCUIL

In paper considered one of decompiling tasks – reconstruction of operators of high-level languages of programming on the example of the object files dcuil.

Keywords: Reverse engineering, CIL, dcuil.

References

1. Aho A. V., Lam M. S., Sethi R., Ulman J. D. Compilers: Principles, Techniques, and Tools. 2nd ed. Moscow, Williams, 2008.

2. Cifuentes C., Simon D., Fraboulet A. Assembly to High-Level Language Translation. Technical Report 439. Department of Computer Science and Electrical Engineering. The Univ. of Queensland, 1998.

3. Cooper K. D., Harvey T. J., Kennedy K. A Simple, Fast Dominance Algorithm. Department of Computer Science, Rice University, 2001.

4. Derevenets E. O., Troshina E. N Strukturnyj analiz v zadache dekompilyacii. [Structure analysis in the decompiling task]. *Prikladnaya informatika*, 2009, no. 4, p. 87–99.

5. Cifuentes C. Structuring Decompiled Graphs. Department of Computer Science, Univ. of Tasmania, 1996.